Training In Register & Micro Operation

- Register Transfer And Micro Operation
- The Multiplexers And Three State Buffer
- Memory Transfer
- Arithmetic Micro Operation
- Design Logic Operation
- Hard Ware Implementation
- Selective – Operations
- Shift Micro Operation

Concepts In Hard Ware

Introduction

ان اى جهاز حاسب ألى هو عبارة عن تصميم معقد من مجموعة اجزاء اليكترونية تعمل مع بعضها بشكل منظم وتلقائى ولكل منها وظيفته الخاصة التى يقوم بها من خلال وضعه فى مكانه الصحيح.

ولكن كيف تعمل هذه الاجزاء مع بعضها البعض بشكل تلقائى؟

ولمعرفة هذا يجب ان نتعرف على ادق تفاصيل الهاردوير . ونجد ان كثير من الناس يعتقدون ان مفهوم الهاردوير قاصر فقط على الاجزاء المادية وان ليس له علاقة بحسابات او اى اجرائات برمجية وهذا فهم خاطىء وذلك لان صناعة الهاردوير اساسا تعتمد على البرمجيات المنطقية وتطوير اساليبها لكى تصبح اكثر تقدما وفاعلية مع احدث البرامج العالمية .

وفى هذا القرص التعليمى سنقوم بشرح مبسط للعمليات التى يقوم بها اى حاسب ألى وكذلك البنية الاساسية له وطريقة الذكاء الاصطناعى التى يفكر بها.

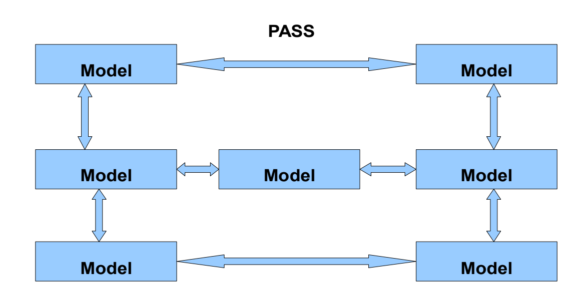
CHAPTER 1

//Register And Transfer//

سنتطرق الان الى مفهوم **Digital System**

اذا اعتبرنا ان اى جهاز **Digital** عبارة عن مجموعة من **Models** مرتبطة مع بعضها بواسطة كابلات تسمى **pass** ونعتبر ان هناك عمليات تنفذ على هذه **Models** وبعد تنفيذها يمكن ان تظل المعلومات التى تحتويها هذه **Models** ان تبقى كما هى او ان تتغير حسب العملية التى اجريت عليها او ان تنتقل عبر **passes** الى مكان اخر مثل **Data Cable** فى الكمبيوتر والذى يحتوى بدوره على ثلاثة انواع من الكابلات :

Data Cable = Data Pass + Address Pass + Control Pass



Digital System

Concepts In Hard Ware

وفيما يلى تعريف لمعنى العمليات :

Operation : هى عملية تنفذ بخطوات محسوبة

Micro Operation : هى جزء من العملية الرئيسية

//Transfer Micro Operation//

هى عملية نقل البيانات من Register الى Register اخر .

: Example : Statement $R2 \leftarrow R1$

Transfer Data In Register R1 To Register R1

ومعانها انتقال البيانات من R1 الى R2 .

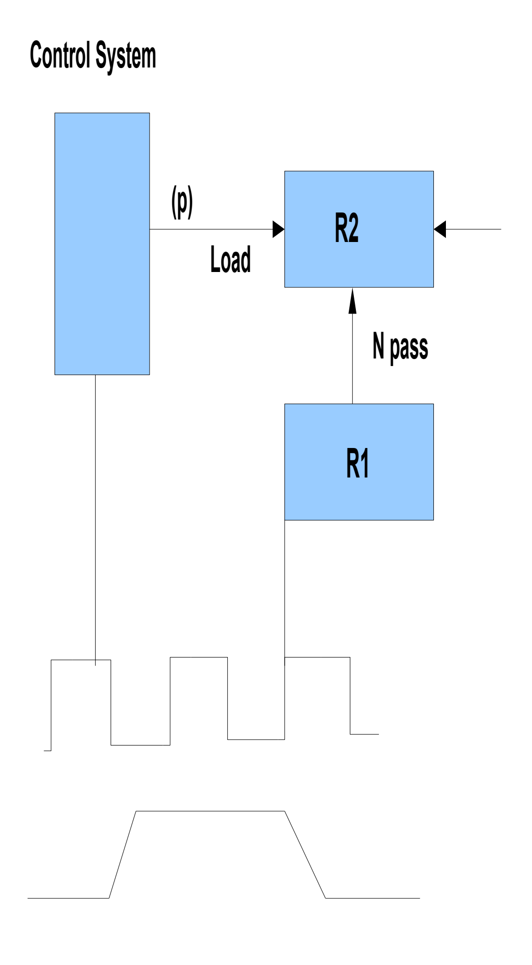
كيفية رسم هذه العملية بواسطة دائرة منطقية .

وتعرف هذه العملية بعملية Timing Diagram

نجد ان Clocking Pulse تلتقى مع Failing Leakage ويحدث Load للعملية اى تحميل لها لكى تنفذ

اذا عندما يكون Load = 1 يحدث Transfer .

Concepts In Hard Ware



Timing Diagram

CHAPTER 2

//Multiplexers & Three State Buffer//

•Multiplexed بأختصار هو عبارة عن Device يستقبل عدد من الدخول ويعطى خرج واحد

كيفية استعمال Multiplexer مع Register لتنفيذ عملية معينة

يحتاج الامر الى بعض الحسابات الرياضية البسيطة لمعرفة عدد Multiplexers التى ستسخدم وعدد الدخول وكيفية اختيار الخرج.

// وسنقوم الان بشرح مثال على ذلك : // لدينا الان عدد Registers ونريد ان نظهر محتويات اى من هما على الشاشة عن طريق وضع الاحتمالات .

Registers are : Register A - Register B - Register C - Register D

: so we can put our calculation

Number Of Registers = 4

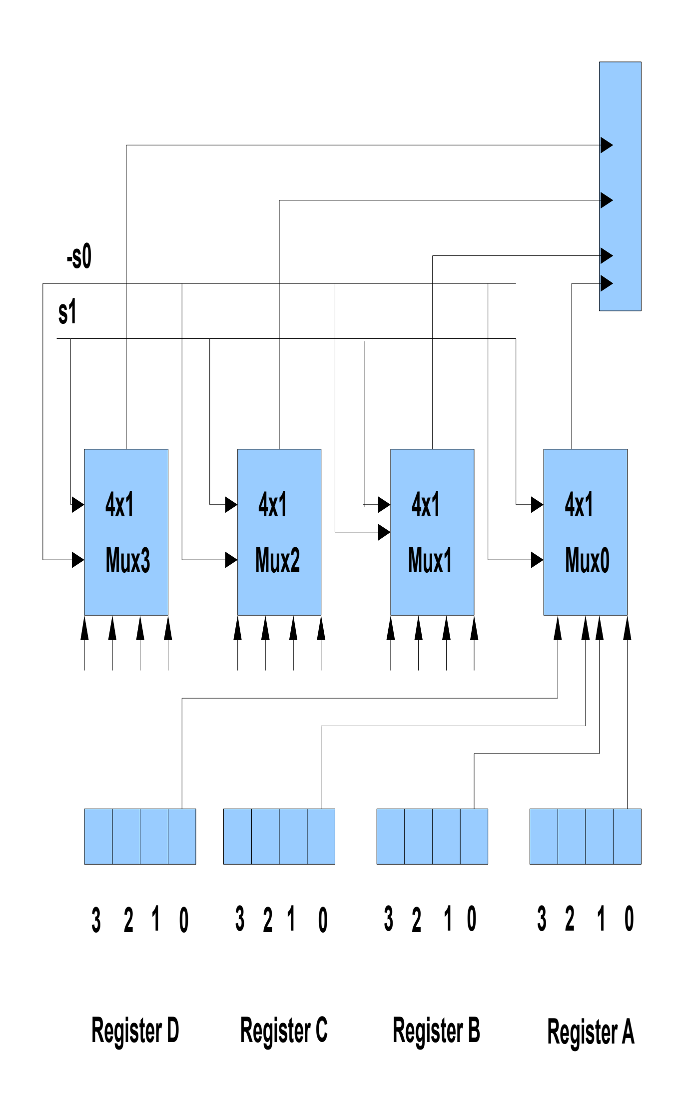
Number Of Inputs = 2 ^ R = 2 ^ 4 = 16INPUT

Number Of Multiplexer = Number Of Bits In Register = 4

Capacity Of Multiplexer = Number Of Register * 1 = 4 * 1 = 4Inputs x 1Output

كيفية رأينا هذه الحسابات لمعرفة عدد ال Multiplexers المستخدمة وكذلك عدد الدخول وسعة Multiplexer .

والان نرى تمثيل عملى لهذه الدائرة وكيفية اختيار Register واحد ليظهر على الخرج .



Bus And Memory Transfer For Four Register

من الرسم نجد لكل نوع من المداخل يتم تخصيص Multiplexer بمعنى :

ان M0 يأخذ الدخول الاتية : (A0 – B0 – C0 – D0)

و M1 يأخذ الدخول الاتية : (A1 – B1 – C1 – D1)

و M2 يأخذ الدخول الاتية : (A2 – B2 – C2 – D2)

و M3 يأخذ الدخول الاتية : (A3 – B3 – C3 – D3)

وبهذا يتم اختيار دخل واحد من كل Multiplexer لنفس Register فمثلا يتم اختيار A0 – A1 – A2 – A3

وبذلك يكون Register رقم A هو الذى يظهر على الخرج وهكذا .

والتمثيل الصحيح لهذه الاختيارات فى علوم الحاسبات يتم عن طريق جداول تسمى Truth Table

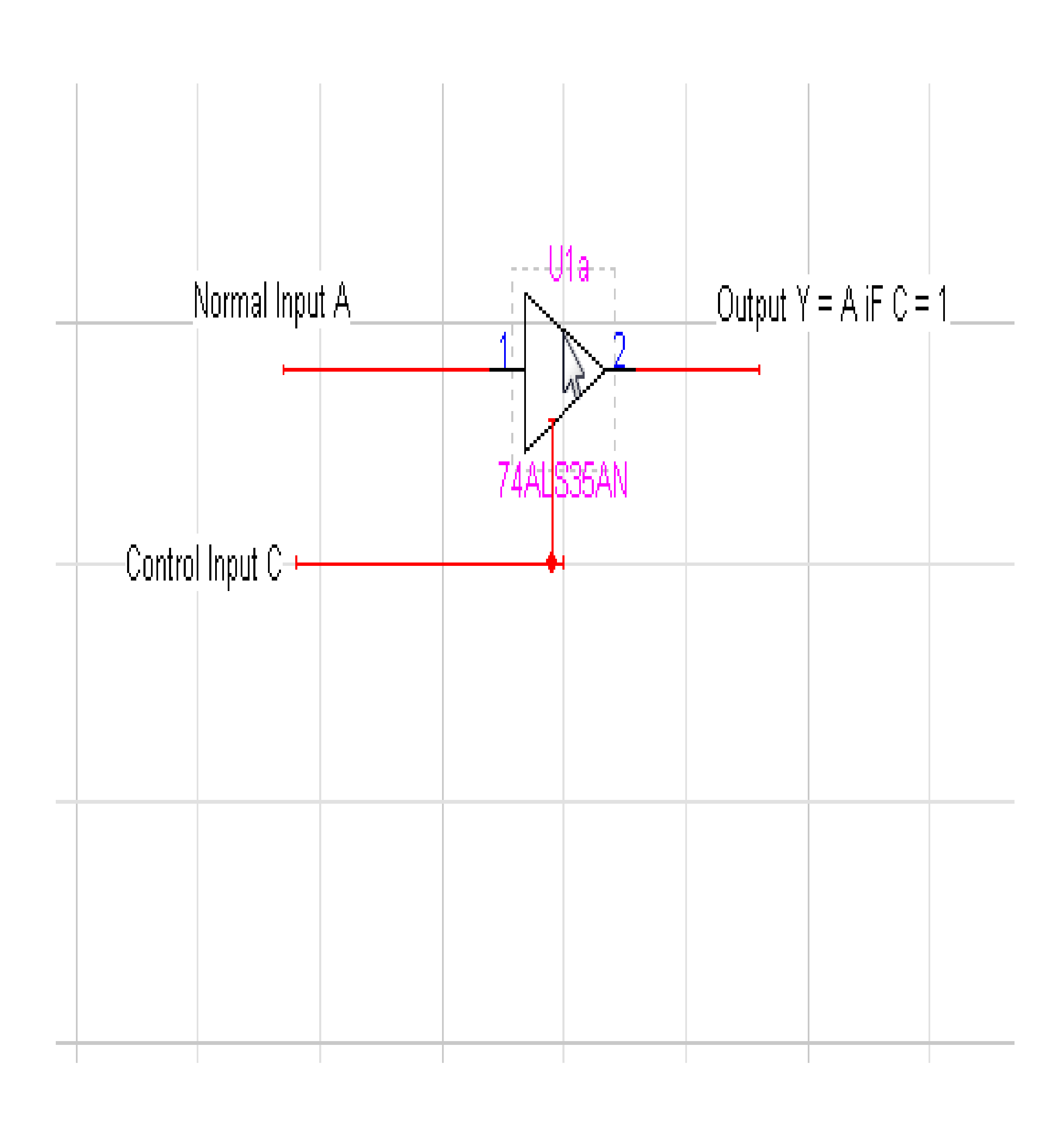
واذا اعتبرنا ان S0 & S1 هما اللذان يقومان باختيار الخرج معا اذا الافتراضات ستكون كالتالى .

S1	S0	Register Selected
0	0	A
0	1	B
1	0	C
1	1	D

//Three – State Bus Buffer//

يعتمد فى هذا الجزء بشكل اساسى على حماية المعلومات وتمريرها كما هى .

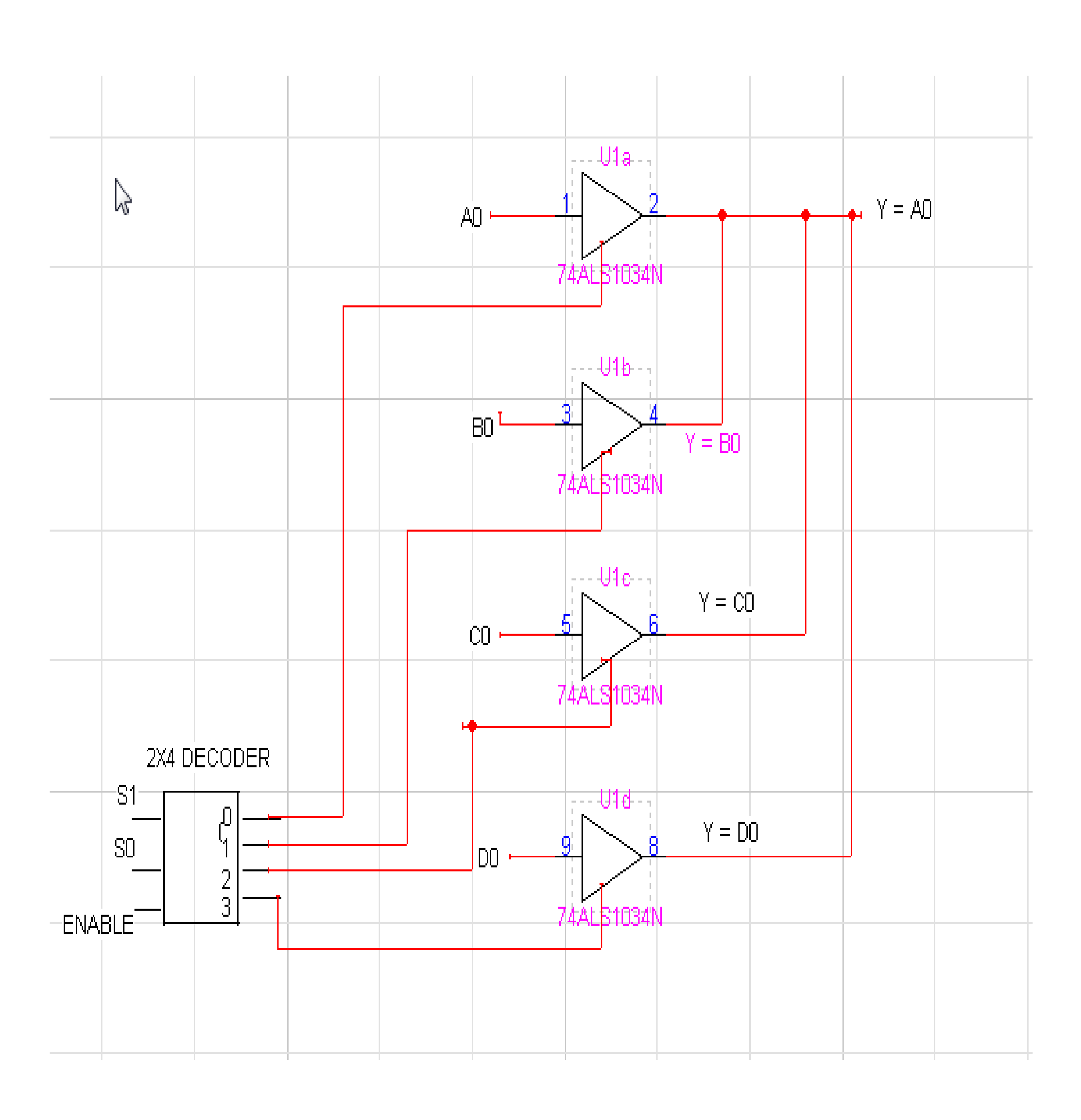
واليك تركيب هذا ال Device



Concepts In Hard Ware

هذا الجزء بأختصار هو عملية حماية للبيانات التى تمر من خلاله ومعنى ذلك ان البيانات التى تمر من خلاله تمر كما هى دون اى تغيير . ويتضح من الرسم ان له ثلاثة اذرع . A ويمثل الدخل . C وتمثل اداة التحكم فى الدخل . Y وتمثل الخرج

والان سنرى كيف يمكن اختيار Register الذى سيظهر على الخرج بواسطة Buffer



Concepts In Hard Ware

فى هذا الشكل نرى كيف تم عمل دائرة بواسطة Decoder & 3State Buffer وذلك لعمل اختيار لاحد Register لكى يظهر على الخرج . فى هذا الشكل نجد ان Register الذى ظهر على الخرج هو Register A وعن طريق رسم نفس الدائرة بواسطة Decoder اخر وتوصيل الكابلات بخط ارضى رئيسى نجد انه يمكن اختيار Register اخر وهكذا .

CHAPTER 3

Memory Transfer

يوجد عمليتين من عمليات انتقال Data من الذاكرة .

.عملية قراءة من الذاكرة (Reading) *** عملية تخزين فى الذاكرة (Writing)**

عملية القراءة (Reading)

Read : Reading operation $DR \leftarrow M[AR]$

ومعنى هذه الجملة انه يتم قراءة البيانات التى توجد فى العنوان المسمى AR فى الذاكرة ووضعه فى DR

عملية الكتابة (Writing)

Write : Writing operation $M[AR]$

ومعنى هذه الجملة ان يتم تخزين البيانات التى توجد فى R1 فى العنوان المسمى AR الذى يوجد فى الذاكرة

CHAPTER 4

//Arithmetic Micro Operations//

هذا النوع من العمليات يطبق على المعلومات التى توجد فى Register وتنقسم الى اربع اجزاء رئيسية .

Transfer Micro Operation From Register To Another

Arithmetic Micro Operation On Numeric Data Stored In Register

Logic Micro Operation On Non – Numeric Data Stored In Register

Shift Micro Operation On Data Stored In Register

عملية نقل المعلومات من Register الى اخر

عمليات حسابية على بيانات زقمية موجودة داخل Register

عمليات منطقية على البيانات الموجودة داخل Register

عملية ازاحة للبيانات من اليمين او اليسار داخل Register الواحد

Basic Arithmetic Micro Operation

Addition – Subtraction – Increment – Decrement – Shift

Contents Of R1 + R2 Transferred TO R3 $R3 \leftarrow R1 + R2$

Contents Of R1 Minus R2 Transferred To R3 $R3 \leftarrow R1 - R2$

S Complement'1 $R2 \leftarrow \bar{R2}$

Concepts In Hard Ware

S Complement'2 $R2 \leftarrow R2+1$

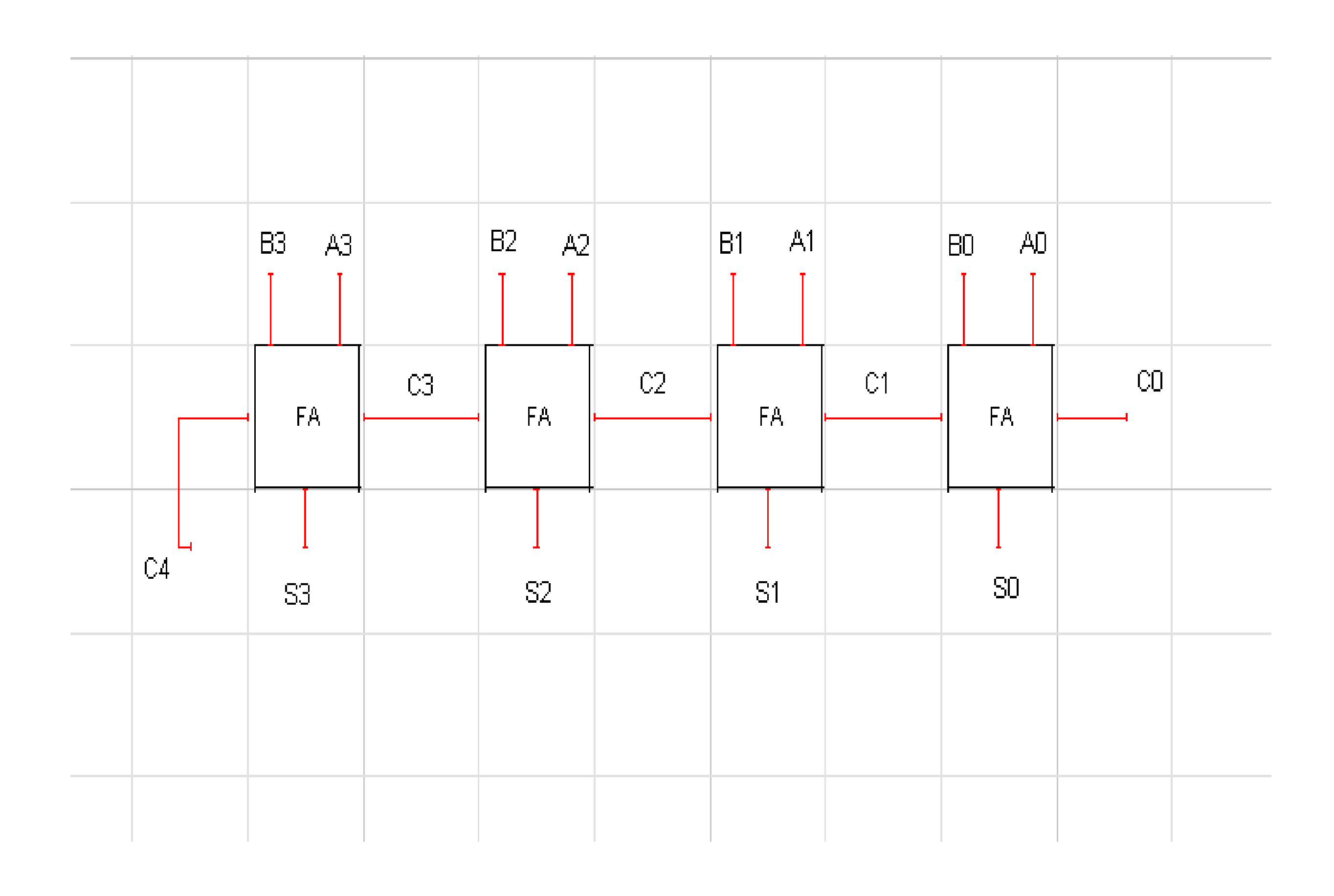
S Of R2 Add R1 Transferred To R3'2 $R3 \leftarrow R1+R2+1$

Increment Of R1 By One $R1 \leftarrow R1+1$

Decrement Of R1 By One $R1 \leftarrow R1-1$

Representation Of Different Operations By Gates

//Binary Adder//



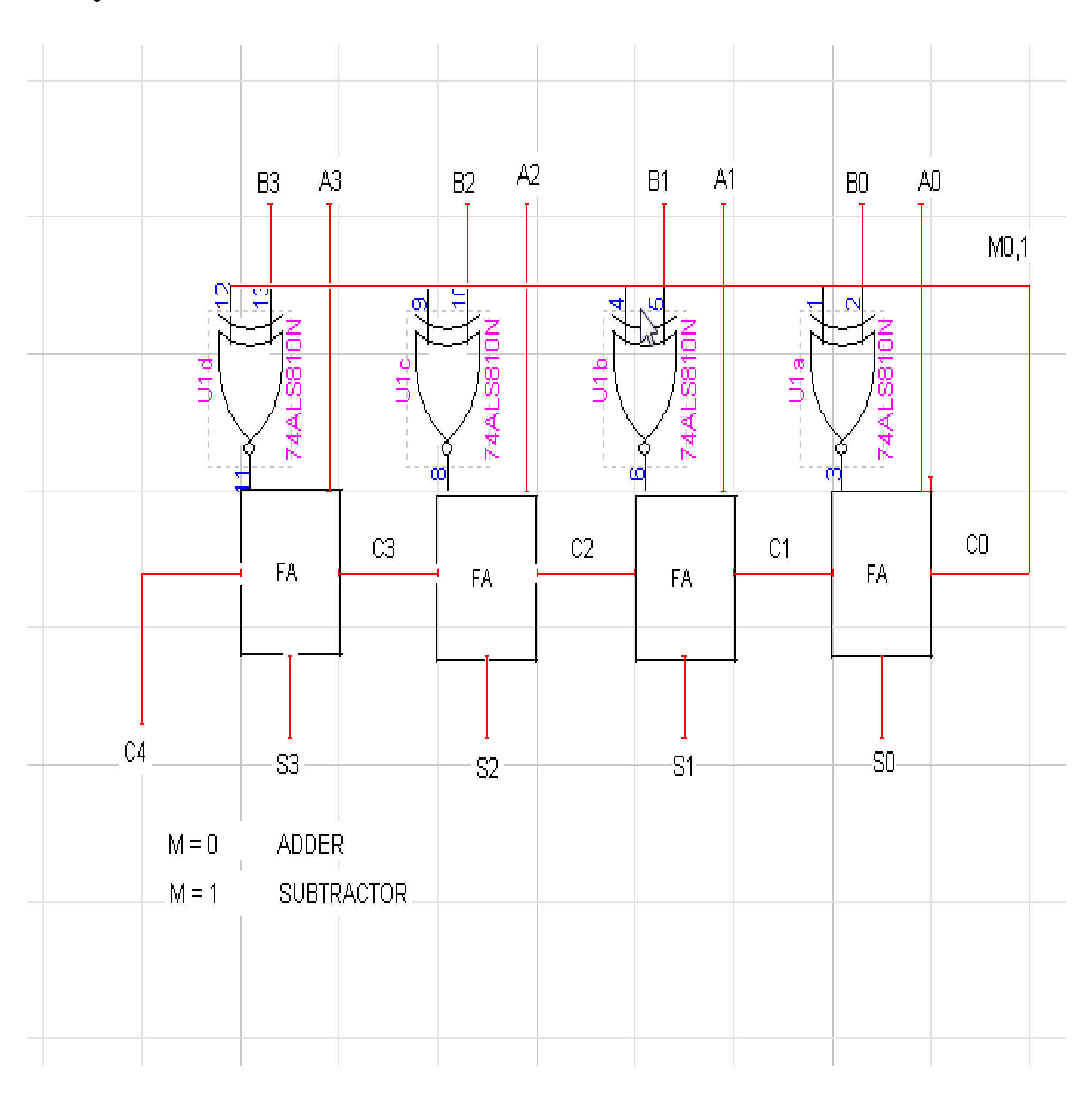
هذه الدائرة بأختصار تمثل Binary Adder وطريقة العمل كالاتى :

// تقوم اول بوابة بجمع اثنين من المدخلات وهما (A0,B0) ثم تضيف على ناتج الجمع ال(CARYY) (C0)

فيكون الناتج S0 ويتبقى من الناتج CARRY اخر C1 يضاف الى عملية الجمع التالية وهكذا حتى تنتهى كل

العمليات على كل البيانات ثم نخرج فى النهاية بناتج جمع S3 , C4 CARRY

//Binary Adder – Sub Tractor//



Concepts In Hard Ware

طريقة عمل هذه الدائرة :

اذا كانت M = 0 :

$A \oplus 0 = B \, , C0 = 0$
$A0 + B0 \, , C0 = 0 \, , S0 = A0 + B0$

اذا كانت M = 1 :

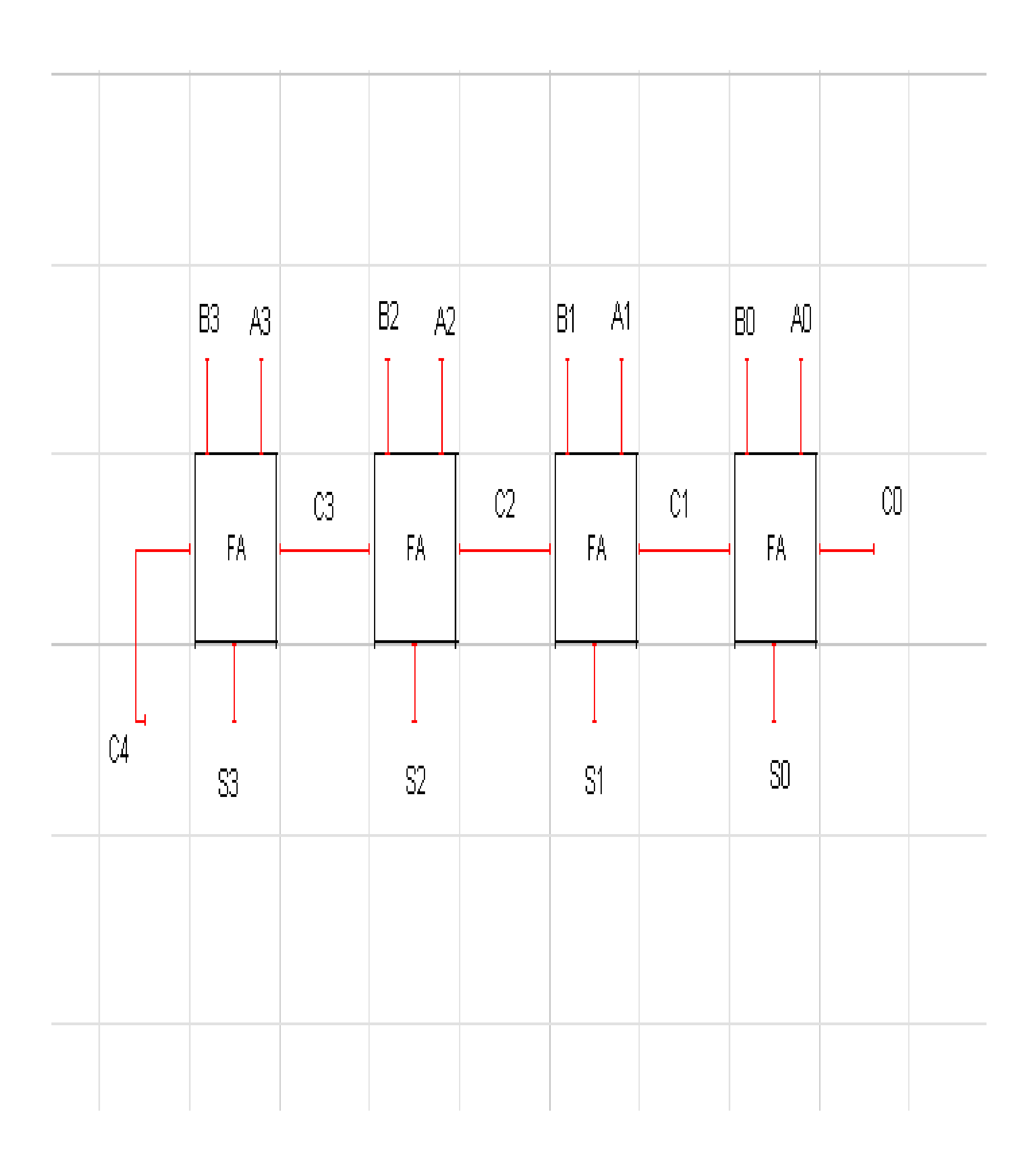
$B \oplus 1 = \overline{B} \, , C0 = 1$

B Is Complemented + 1 **2's Complement**

A + 2'S Complement B **A – B,C0 = 1**

رأينا كيف استطعنا بهذه الدائرة ان نقوم بالعمليتين معا جمع وطرح فعندما تكون الاختيار = 0 نجد ان العملية تكون جمع وذلك لان الصفر يمرر الناتج كما هو خلال بوابة XOR اما عندما يكون الاختيار = 1 فنجد ان العملية تتحول الى طرح وذلك لان الناتج يكون مع الواحد الصحيح خلال هذه البوابة هو بمثابة Complement للرقم

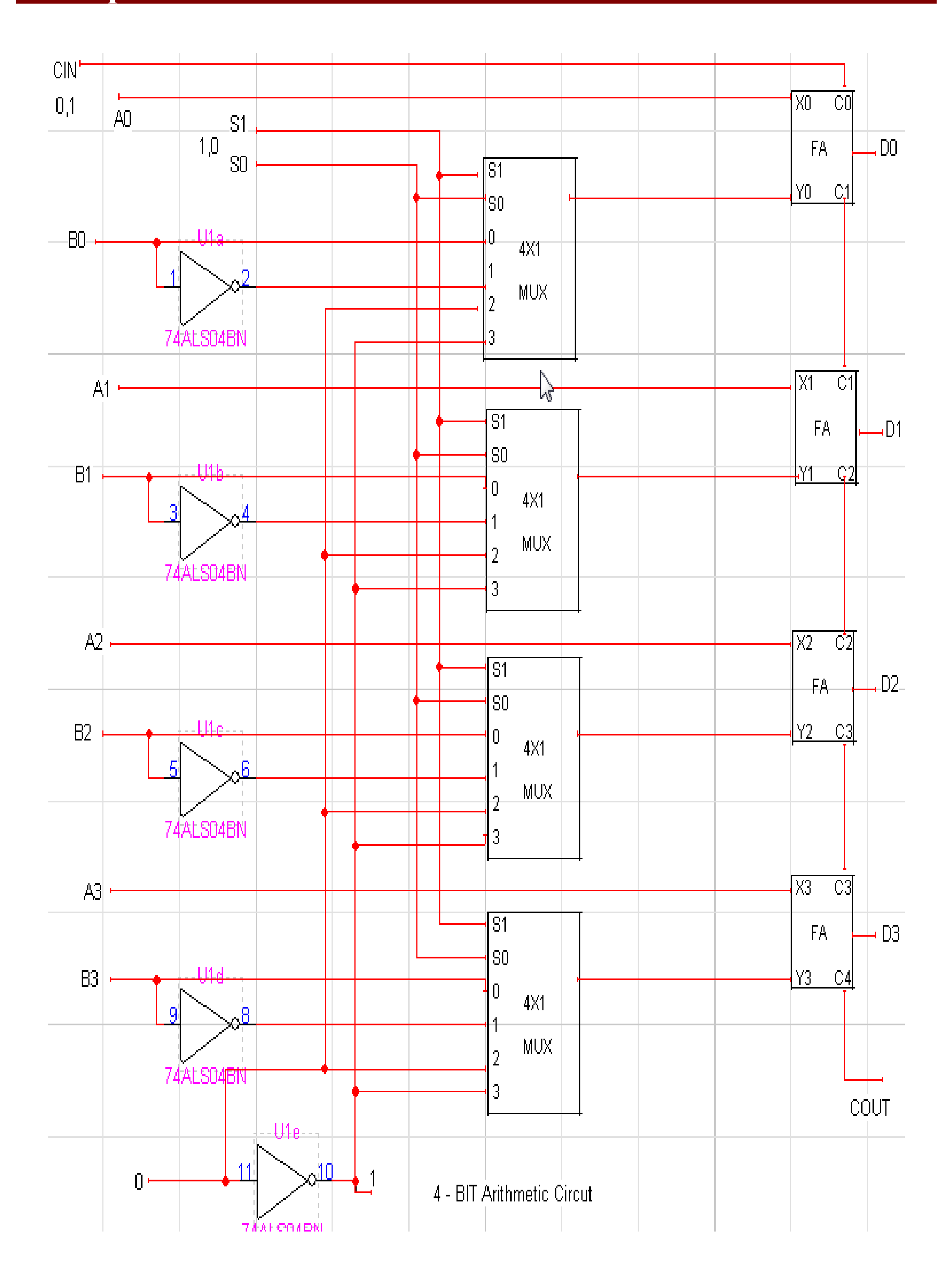
//Binary Incremented//



تبداء عملية الجمع بأضافة 1 على A0 ثم ينتج S ويتبقى C1 ويكون دخل جديد لعملية الجمع التالية مع A1 وهكذا حتى ننتهى فتكون العملية بمثابة اضافة 1 على ال Register

//Arithmetic Circuit//

ان القاعدة التركيبية لهذه الدائرة هى مجموعة من بوابات الجمع Adder Gates موصلة مع بعضها على التوازى ويوجد تحكم فى عدد ونوع Inputs وبذلك نستطيع ان نحصل على عدد مختلف من العمليات الرياضية المختلفة وتعتبر هذه الدائرة الاكثر تعقيدا .


CIN
0,1
A0
S1
1,0
S0
B0
U1a
74ALS04BN
4X1
MUX
A1
B1
U1b
A2
B2
U1c
A3
B3
U1d
U1e
0
1
FA
D0
D1
D2
D3
COUT
4 - BIT Arithmetic Circut

شرح عمل الدائرة :

يوجد 4 دخول لهذه الدائرة تطبق على عدد 4Bites فقط ويوجد 4 بوابات Full Adder

كل دائرة تحتوى على 4Bites و 4Multiplexer وذلك لاختيار العمليات المختلفة وعدد الدخول 4Inputs

ل A,B و 4Bites خرج وهى D

4 دخول ل A يذهبون مباشرتا ل Input x بالنسبة لدوائر Adder .

كل دخل من B يذهب الى Data Input لل Multiplexer . ودخل B مباشرتا ونفيه الى Multiplexer

الدخلين 3,2 موصلين ببوابة logic 0,1

logic 0 ثابتة للدائرة كلها .

Logic 1 يولد من عكس Logic 0

التحكم فى اختيار Multiplexer عن S0,S1

ال CARRY يضاف الى بعضه على التوالى

خرج كل Binary Adder يحسب كالتالى D = A + Y + CIN

بواسطة التحكم فى S0,S1 , CIN1,0 يتولد 8 عمليات رياضية .

//Eight Arithmetic Micro Operations//

S1	S0	CIN	INPUT Y	OUTPUT D = A + Y + CIN	MICRO OPERATION
0	0	0	B	$D=A+B$	ADD
0	0	1	B	$D=A+B+1$	ADD WITH CARRY
0	0	0	$\bar{B}$	$D=A+\bar{B}$	SUBTRACT WITH BORROW
0	1	1	$\bar{B}$	$D=A+\bar{B}+1$	SUBTRACTION
1	0	0	0	$D=A$	TRANSFER A
1	0	1	0	$D=A+1$	INCREMENT A
1	1	0	1	$D=A-1$	DECREMENT A
1	1	1	1	$D=A$	TRANSFER A

كيفية تكوين العمليات :

عندما تكون الاختيارات S0,S1 = 00 نجد ان الدخل سيكون = B وعندما يكون CIN = 0 اذا يكون الخرج على هذه الصورة .

D = A + B

ولكن اذا كان CIN = 1 فاءن الخرج يكون على هذه الصورة .

D = A + B + 1

وكلاهما تعتبر عملية جمع ولكن الاولى Without Carry والثانية With Carry

عندما تكون الاختيارات S0,S1 = 0,1 نجد ان Complement of B = B هى التى تطبق على الدخل وعندما يكون CIN = 1 يكون الخرج على هذه الصورة .

$D=A+\bar{B}+1$

وهذا يعنى جمع A مع S'2 لل B وهذا بمثابة طرح فيكون الخرج النهائى =

D = A – B

ولكن عندما يكون CIN = 0 فيكون الخرج =

D = A Transfer Operation

تعتبر عملية انتقال ال A الى الخرج

اما اذا كان CIN = 1 فاءن الخرج

D = A + 0 + 1 Increment To A By 1

عندما تكون الاختيارات S0,S1 = 1,1 و CIN = 0 سيكون الخرج على هذه الصورة .

D = A – 1

عندما تكون CIN = 1

D = A – 1 + 1 = A Direct Transfer

عملية انتقال مباشرتا للخرج .

لاحظ ان عملية الانتقال المباشر قد تمت مرتين و لهذا فاءن Arithmetic Circuit ينتج عنها 7 عمليات مختلفة وعمليتين متشابهتين .

CHAPTER 5

Logic Micro Operations

العمليات المنطقية نجد انه فى سياق تلك العمليات تعتمد بشكل اساسى على التفكير المنطقى وليس الرياضى .

وسنرى المثال التالى :

لدينا الان **R1 = 1010 , R2 = 1100**

والمراد اجراء عملية **XOR** وارسال الناتج الى **Register R1**

$P: R1 \leftarrow R1 \oplus R2$

1010
1100

0110

ولاحظ ان $A \oplus 1 = \bar{A}$, $A \oplus 0 = A$ وال **P** هى بمثابة الاشارة التى تعطى امر تنفيذ العملية اذا كانت

= 1 العملية تنفذ واذا كانت **= 0** لا تنفذ .

والان لدينا بعض اشكال العمليات المنطقية :

$P + Q: R1 \leftarrow R2 + R3$

P + Q تسمى **OR Operation** ، و **R2 + R3** تسمى **Add Micro Operation**

$R4 \leftarrow R5 \vee R6$

$R5 \vee R6$ تسمى **OR Micro Operation**

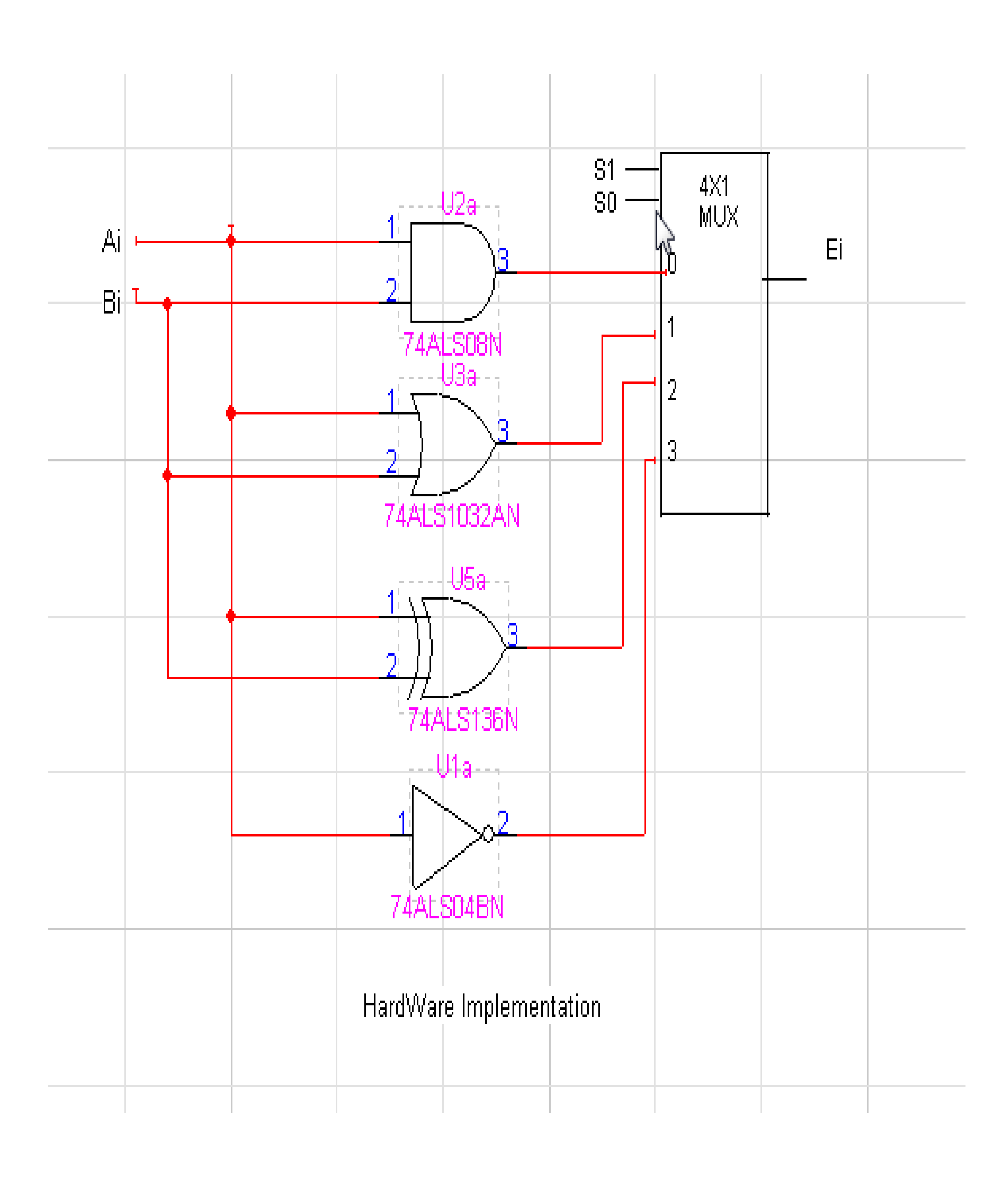
والان سنقوم بسرد جدول العمليات المنطقية الشائعة وكذلك جدول الاحتمالات لها :

// List Of Logic Micro Operations//

x	Y	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1
1	1	0	1	0	0	0	0	0	0	0	1	0	1	0	1	0	1

BOOLEAN FUNCTION	MICRO OPERATION	NAME
$F0=0$	$F \leftarrow 0$	CLEAR
$F1=XY$	$F \leftarrow A \wedge B$	AND
$F2=X*\bar{Y}$	$F \leftarrow A \wedge \bar{B}$	
$F3=X$	$F \leftarrow A$	TRANSFER
$F4=\bar{X}*Y$	$F \leftarrow \bar{A} \wedge B$	
$F5=Y$	$F \leftarrow B$	TRANSFER B
$F6=X \oplus Y$	$F \leftarrow A \oplus B$	EX - OR
$F7=X+Y$	$F \leftarrow A \wedge B$	OR
$F8=(X \overline{+} Y)$	$F \leftarrow (A \overline{\wedge} B)$	NOR
$F9=(X \overline{\oplus} Y)$	$F \leftarrow (A \overline{\oplus} B)$	EX - NOR
$F10=\bar{Y}$	$F \leftarrow \bar{B}$	COMPLEMENT B
$F11=X+\bar{Y}$	$F \leftarrow A \wedge \bar{B}$	
$F12=\bar{X}$	$F \leftarrow \bar{A}$	COMPLEMNT A
$F13=\bar{X}+Y$	$F \leftarrow \bar{A} \wedge B$	
$F14=(\overline{XY})$	$F \leftarrow (A \overline{\wedge} B)$	NAND
$F15=1$	$F \leftarrow 1'S$	SET ALL 1'S

//Hard Ware Implementation //



HardWare Implementation

S1	S0	OUTPUT	OPERATION
0	0	$E = A \wedge B$	AND
0	1	$E = A \vee B$	OR
1	0	$E = A \oplus B$	XOR
1	1	$E = \bar{A}$	COMPLEMENT

//Selective – Set Operations Sets//

1010
1100

1110

A = 1010 وهذا يمثل الرقم قبل العملية . A Before

B = 1100 وهذا يمثل المعامل . Logic Operand

لاحظ ان الرقمين على اليسار من B = 1's للرقمين على يمين ولذلك يتم تغيير هذه الارقام التى توجد على يسار ال A الى 1 فيصبح الناتج = 1110 ونجد ان الرقمين على يمين B = 0 اذا الرقمين على يمين A يضلوا كما هم .

//Selective – Complement Operation//

1010
1100

0110

نفس العلامة السابقة مع اختلاف هو ان نحضر 1's للرقمين على اليسار من B .

//Selective – Clear Operation//

1010
1100

0010

اذا كان الرقمين على اليسار من B = 1's للرقمين على اليمين اذا الرقمين على لايسار من A يتم وضعهم

= 0 وتسمى العملية Clear

//Mask Operation//

1010
1100

1000

الرقمين على يمين A يتم وضعهم = 0 وذلك لان الرقمين التابعين لهم فى B = 0's لهم

وسمى هذه العملية بعملية القناع Masking

اما الرقمين على يسار A لم يتم تغيرهم وذلك لان الارقام التابعة لهم فى B = 1's لهم .

//Insertion Operation//

وتعتبر من اهم العمليات المنطقية اذ تكمل على مرحلتين .

المرحلة الاولى .

01101010
00001111

00001010

لاحظ ان ال A هو الرقم و B هو Mask اذا المراد اولا عملية Masking لل A

نريد حذف الاربع ارقام على يسار A وتستبدلهم بأرقام جديدة ولذلك تم عمل Mask لهم وذلك بوضع اربع

ارقام اصفار 0000

اما الارقام على يمين A تم وضع تحتهم 1's حتى يظلوا كما هم .

Insert The New Value

بعد ذلك يتم عمل Insertion للقيمة الجديدة

00001010
10010000

10011010

ولاحظ هنا ان العملية عملية جمع .

عندما نريد ان نختبر اذا كان القيميتين متساويتين نعمل لهم EX – OR حيث يكون الناتج الكلى = 0

1010
1010

0000

//Shift Micro Operations//

الانواع :

(Logic Shift Micro Operations)

(Circular Shift Micro Operations)

(Arithmetic Shift Micro Operations)

$R \leftarrow SHL\ R$ **Shift – Left**
$R \leftarrow SHR\ R$ **Shift – Right**
$R \leftarrow Cil\ R$ **Circular Shift – Right**
$R \leftarrow Cir\ R$ **Circular Shift – Right**
$R \leftarrow ASHL\ R$ **Arithmetic Shift – Left**
$R \leftarrow ASHR\ R$ **Arithmetic Shift - Right**

Shift معناها الازاحة والعمليات ببساطة هى اما ازاحة الرقم يمينا او يسارا او تدويره ثم ازاحته .

CHAPTER 6

//Basic Computer Organization//

............................Stored Program Organization

..... Computer Registers

Basic Registers..
Common Bus System..

..... Computer Instruction

Instruction Format..
Instruction Set..

الان نبداء فى شرح البنية الاساسية لاى جهاز كمبيوتر :

ونبداء بتعريف عملية Register Transfer : وهى لغة تستخدم فى وصف العمليات التى تحدث داخل الحاسب

وتتوقف عليها عملية تصميم الهاردوير داخل الحاسب .

كيفية سير المنظومة الداخلية لعمل الحاسب الالى :

.كيف يتم تنظيم خطوات برنامج داخل الحاسب لتشغيله .

الحاسب يتعامل فقط مع لغة Binary اى 0,1 وهذه لغة لايستطيع المستخدم ان يفهمها فما الحل .

.الصورة العامة *

اى عملية يتم تنظيمها من خلال برنامج يقوم المستخدم بوصفه

Concepts In Hard Ware

وتسمى اللغة العالية High Level Language

Programmer............. High Level Language

اى اللغة التى يفهمها الحاسب .

اما اللغة التى يفهمها الحاسب .

Low Level Language................Machine

ويتم التحويل بينهما عن طريق Compiler

High Level Language......... Compiler ....Assembly............Low Level Language

مثال توضيحى .

$HLL \rightarrow LLL , 5+6 \rightarrow ADD 5 , 6$

نرى الان كيف يتم تحويل لغة الامستخدم الى لغة تفهمها الالة . From 5 + 6 to ADD 5,6

الان نرى كيف التحويل بين اللغة التى يفهمها الحاسب والتى يفهمها المستخدم فى الشكل اسفل

نحدد العملية والمعاملات فالعملية هى جمع وكود الجمع الذى يفهمه الحاسب هو 101 والمعاملات او Data

هى 5,6 والكود الذى يفهمه الحاسب هو 101 = 5 و 110 = 6

Assembly ---------Assembler ----------------Machine Code

اما اذا كنا سنتعامل مع متغيرات يجب حجز مكان فى الذاكرة كالتالى .

$X+Y \rightarrow ADD X , Y$

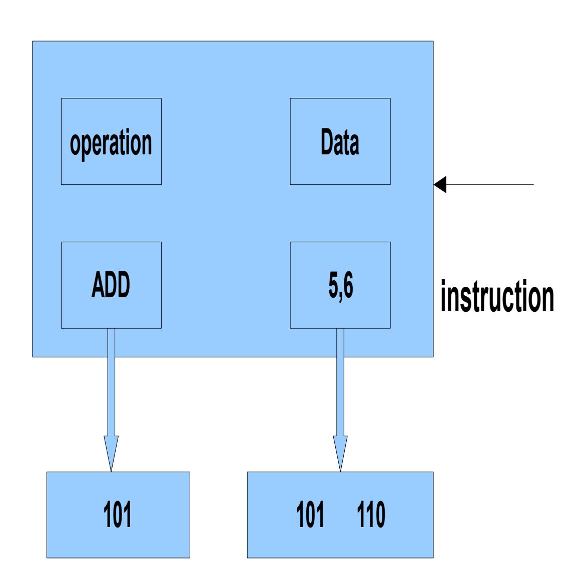
X هو عنوان مكان فى الذاكرة و Y هو عنوان مكان اخر حيث يتم جمع المحتويات بداخلهما .

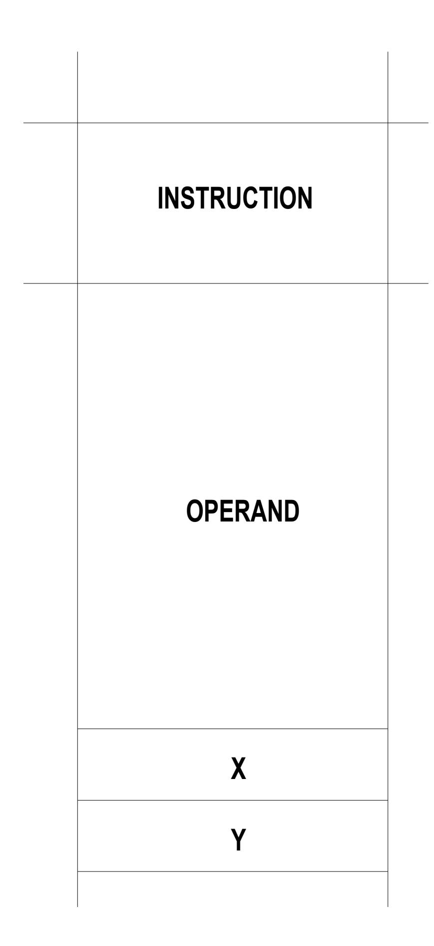
اذا اى Instruction يجب تحويلها الى بواسطة Assembly الى Machine Code لكى يفهمها الحاسب

وبعد ذلك يتم تخزينها فى الذاكرة Main Memory

وعملية التخزين تتم على مرحلتين :

Concepts In Hard Ware





الان نرى تركيبة اى عملية داخل Register

Instruction هى اوامر العمليات و operand هو المعامل و x,y هما Data1,Data2

.مرحلة التنفيذ تتم على ثلاثة مراحل :.

1- Fetch instruction Register

وهى تقوم بعملية نقل Instruction الى Register داخل المعالج

Decod Instruction – 2

عملية يقوم بها Control Unit ويتم من خلالها تحديد العملية الموجودة داخل Register وتحدد من خلالها جزء العملية داخل Instruction

Execute – 3

تنفيذ العملية بعد تحديدها من خلال Arithmetic Logic او Arithmetic Unit

اذا العمليات تتم على مرحلين اساتين :

1 – نقل البيانات الى مجموعة من Registers

2 – تنفيذ العملية بواسطة Alu ومحتوى Register وبعد ان تتم العملية يتم اعادة تخزينها فى Memory

ملاحظة : - لا يمكن تنفيذ اى عملية داخل Memory ولكن يتم نقلها الى Register وتنفيذها بواسطة Alu واعادتها الى Memory

اذا منظومة الحاسب الالى تعتمد بالاساس على

1 – مجموعة من Register و Control And Timing Unit لتحديد نوع العملية ومتى تنفذ

2 – مجموعة التعليمات التى يعمل عليها الحاسب الالى والتى تسمى Instruction

التعليمات لا تختلف من جهاز الى جهاز بشرط ان تكون من نفس العائلات :

مثلا : -

AppleMachintouse ابل ماكينتوش لها مجموعة اوامر خاصة بها

Motorola موتورلا لها مجموعة اوامر خاصة بها

ولكن من ناحية Software فهو المسئول عن عمل Compile لاى برنامج والذى يختلف هى كيفية تمثيل العملية من جهاز لاخر

مجموعة Register الاساسية التى توجد داخل اى حاسب الى والتى يبنى عليها عمل الحاسب : -

Concepts In Hard Ware

ـفى الحقيقة نجد ان اى حاسب الى بل ابسط حاسب الى يجب ان يحتوى على 8Registers هذا بالاضافة الى Main Memory واليك التمثيل التالى للبنية الاساسية للكمبيوتر :

Main Memory

Data Register

Accumulator Register

PC

DR

AR

AC

IR

TR

OUT.R

INP.R

Concepts In Hard Ware

وفيما يلى سنرى فائدة كل Register ودوره فى اداء مهمة للكمبيوتر :

Data Register : يحتوى على اى بيانات مخزنة فى Memory او بيانات يتم قرائتها من Memory

اذا اى بيانات تقراء او تكتب الى Memory يتم نقلها اولا الى DR

وبناء على ذلك فاءن حجم DR = حجم Word داخل Memory .

و memory هى عبارة عن مجموعة من Registers متتالية مع بعضها البعض .

Address Register : يشير الى عنوان مكان فى الذاكرة ليتم التعامل مع هذا المكان .

اذا لتعامل مع اى مكان يجب تحميله اولا الى AR وهو يشير الى مكان واحد ويحدد عدده حسب عدد الاماكن الموجودة فى الذاكرة .

اذا كان حجم الذاكرة مثلا = 2^N اذا عدد AR = N

اذا كان حجم الذاكرة = 4 = 2^2 اذا عدد AR = 2Bits

اذا كان حجم الذاكرة = 16 = 2^4 اذا عدد AR = 4Bits

ومن ذلك نستخلص ان هناك نوعين من R يتعاملوا مع Memory وهما AR,DR

Programmer Counter : يحمل عناوين اماكن داخل الذاكرة وهو عنوان المكان التالى لعملية التنفيذ .

عند تنفيذ البرنامج لاول مرة سيتم نقل اول عنوان الى PC ثم ينتقل الى AR لينفذ ثم PC يعمل عمل Increment اضافة واحد وهو الخطوة التالية للعملية التى تنفذ حاليا وبالتالى يعمل البرنامج اوتوماتيكيا

ونجد ان حجم PC = حجم AR

Instruction Register : يحتوى على Instruction الجارى تنفيذها

AR---------> يشير الى عنوان مكان فى الذاكرة -----> ينتقل الى-----> DR------>ينتقل الى ------>IR

حجم IR = حجم Memory

Concepts In Hard Ware

Temp Register : يحتوى على Data مؤقتة لاجراء بعض العمليات المعينة وحجم TR = حجم Memory

Input Register : يستخدم لتخزين Input Character ممكن ان يكون حرف او رقم = Data

وحجمه كاقصى حد = 8Bites

Output Register : ويحتوى على Output Character وحجمه = 8Bites

Accumulator Register : يحتوى على اول Data تجرى عليها العملية ويسمى Processing Register

For Example :

ADD 5,6 DR = 6 AC = 5

وبعد ذلك تتم عملية الجمع بواسطة ALU

//Computer Instruction//

Instruction Types : انواع العمليات

Instruction Formate : نمط العملية

Instruction Cycle : دورة العملية

//Instruction Types//

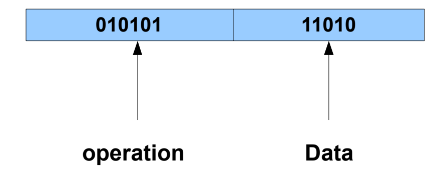
1 – Register Reference Instruction :

2 – Input / Output Reference Instruction :

Instruction هى فى الاساس عبارة عن مجموعة من Binary Bites وتمثل جزئين جزء يمثل نوع العملية واخر يمثل البيانات التى تتم عليها العملية .

Concepts In Hard Ware

وكل عملية محددة بكود يمثلها



Data يمكن ان توجد فى الذاكرة او داخل Internal Register او IP/OP Register وكل نوع له تعامل

خاص وبناء على مكان Data يتم تصنيف نوع العملية

Memory Reference – 1

Register Reference – 2

IP/OP Reference – 3

Concepts In Hard Ware

Examples
:

ADD	ADD1,ADD2
INC	AC
INP	TR
OUT	OUTR

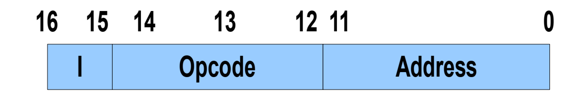
Concepts In Hard Ware

ولكن كيف يتمكن الحاسب من تحديد نوع **Instruction** وبالتالى تنفيذها

اى حاسب الى يستطيع من جزء داخل **Instruction** تحديد نوع الخطوة وبالتالى تحديد مكان **Data** وهذا الجزء يسمى **Operation Code**

الشكل العام للعملية يتكون من ثلاثة اجزاء :

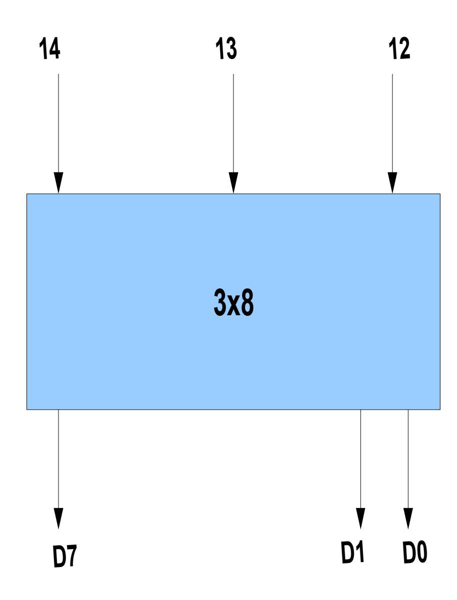
Instruction Formate



كما نرى الثلاثة اجزاء الرئيسية هما :

Mode Bit – Opcode – Address

ونجد ان محتوى Opcode ينتقل الى Decoder لكى يتم تحديد نوع الخطوة التى ستنفذ ونوع العملية



وظيفة كل جزء من اجزاء العملية :

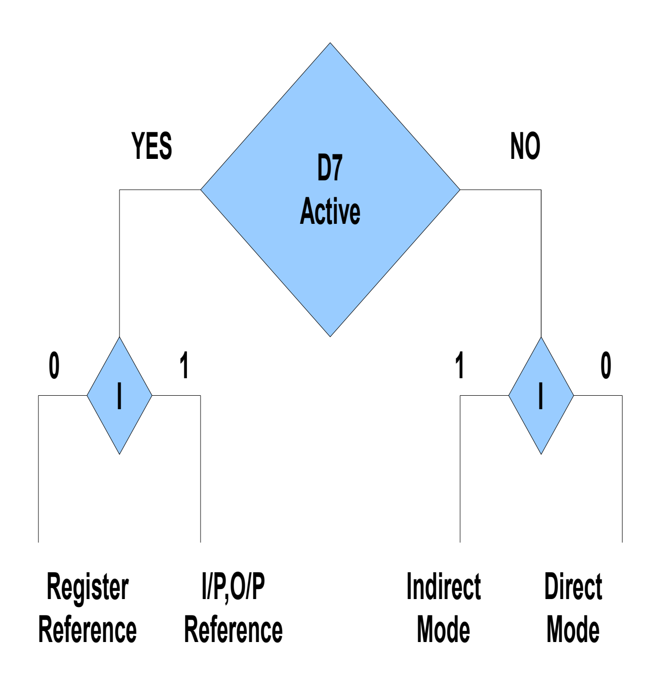
(Operation Code) : والذى يختصر الى Opcode

يقوم بتحديد ما اذا كانت نوع العملية I/P ,O/P Reference او Register Reference

والذى يحدده I

اذا كان I = 0 تكون Register Reference واذا كان I = 1 تكون I/P,O/P Reference

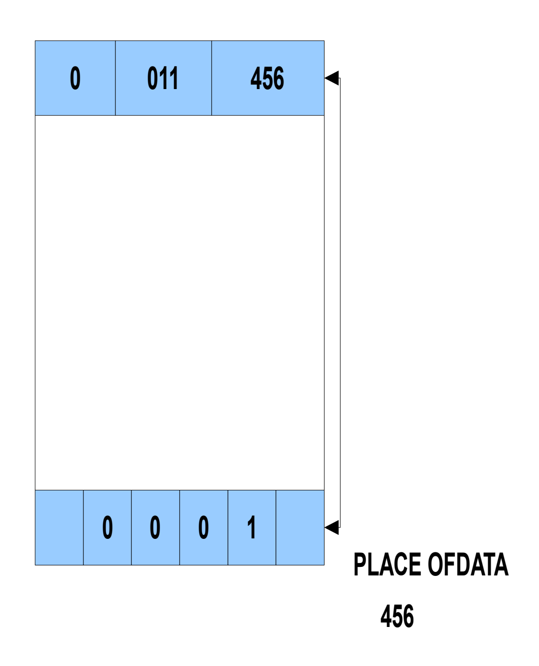
ونبداء اولا باختبار D7 والذى يتضح فى الرسم التالى :



YES
NO
D7
Active
0
I
1
1
I
0
Register
Reference
I/P,O/P
Reference
Indirect
Mode
Direct
Mode

: Direct Operation

فى هذا النوع من العمليات نجد ان Address Filed يشير الى مكان وجود البيانات ولذلك تسمى بالعملية المباشرة .

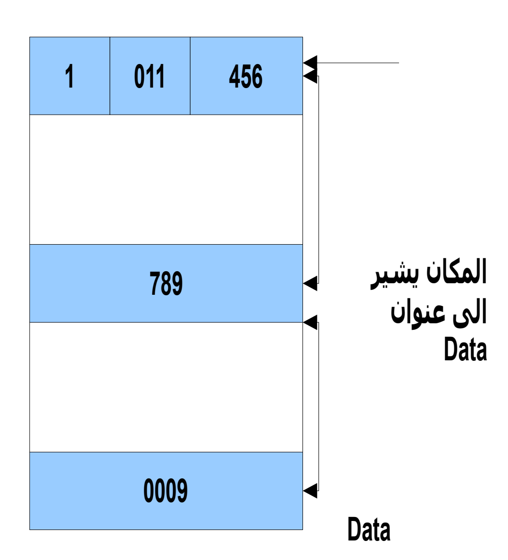


نجد ان Address Filed اشار مباشرتا

الى عنوان Data

: Indirect Operation

هنا نجد ان Address Filed يحتوى على عنوان مكان هذا المكان يحتوى على عنوان Data



نجد هنا ان 456 قد اشار الى عنوان مكان هذا المكان هو 789

هذا المكان اشار الى عنوان البيانات وهى 0009

Concepts In Hard Ware

اذا الاختلاف هو الوصول الى عنوان البيانات بطريقة مباشرة او غير مباشرة ويحدد نوع العملية و Register الذى تجرى عليه العملية على اساس Bites الموجودة فى Address .

كيفية تنفيذ الخطوة داخل البرنامج : Instruction Cycle

مراحل التنفيذ :

Fetch :

فى هذه المرحلة يتم انتقال البيانات من الذاكرة الى Instruction Register

Decode :

يتم تحديد نوع العملية – تحديد البيانات

Execute :

يتم تنفيذ العملية على البيانات

الشيفرات التى تشير الى مراحل التنفيذ :

Fetch :

$$T0: AR \leftarrow PC$$

$$T1: IR \leftarrow M[AR], PC \leftarrow PC+1$$

: Decode

تحليل محتوى الخطوة الموجودة فى AR

$$T2: Decoder \rightarrow IR(12, 13, 14), AR \rightarrow IR(0 \rightarrow 11), I \rightarrow IR(15)$$

نجد ان 12و 13و 14 يتم عن طريق ال Decoder تحديد نوع العملية

0 الى 11 عن طريق AR يشير الى عنوان Data

15 بتحديد I يتحدد نوع ال Register

: Execute

مرحلة التنفيذ وتعتمد على نوع العملية :

والذى يحدد العملية Address Field ويوجد فيه Bit واحد فقط يحدد نوع العملية ونوع Register الذى تجرى عليه العملية

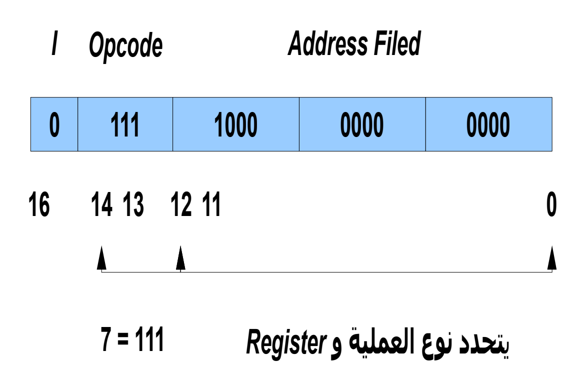
$$T3\ D7\ \bar{I}: EXecute\ Register\ Reference$$

$$T3\ D7\ I: Execute\ IP/OP\ Register\ Reference$$

$$T3\ \bar{D}7\ \bar{I}: Read\ Data\ Direct$$

$$T3\ \bar{D}7\ I: Read\ Data\ Indirect$$

والان ناتى لكيفية تنفيذ Register – Reference Instruction



Concepts In Hard Ware

كيفية كتابة كود العملية :

القواعد : اى خطوة يبداء من اليسار ب 7 تكون Register – Reference Instruction ويتبقى الترتيب الكودى

يتم تحديد العملية و Register الذى تجرى عليه العملية عن طريق Bit واحد فقط يكون هو Active والباقة OFF ولاحظ ان Opcode , I ثابتين وبالتالى يتولد 12 عملية مختلفة .

لاحظ ان خط السير يبداء الان من اول Bit على اليسار الى اخر Bit على اليمين يتم تنقيل الواحد الصحيح من اليسار الى اليمين حتى يتولد 12 عملية مختلفة .

1 - عملية Clear وهى تحويل محتوى AC الى 0

ونلاحظ ان T3 D7 BAR I ثابتة لجميع العمليات اى من I وال Opcode الى BIT رقم 12 ثابتين

ولذلك سنعوض عنهم فى كل العمليات التالية ب R

اذا R = T3 D7 BAR I

الشفرة :

$$CLEAR \rightarrow CLA \rightarrow 7800$$
$$7800\ CLA\ RB11 : AC \leftarrow 0$$

2 – عملية CLE وتسمى Extended Flip Flop = 0

هذا النوع يوضع فيه Carry , Carry out , Over

الشقرة :

$$EXTENDED\ FLIP\ FLOP \rightarrow CLE \rightarrow 7400$$
$$7400\ CLE\ RB10 : E \leftarrow 0$$

3 – عملية CMA تحويل S'1 الى محتوى AC

الشفرة :

$$CMA \rightarrow 7200$$

$$7200\ CMA\ RB9 : AC \leftarrow \bar{AC}$$

4 – CME تحويل S'1 الى محتوى Flip Flop

الشفرة :

$$CME \rightarrow 7100$$

$$7100\ CME\ RB8 : E \leftarrow \bar{E}$$

5 – CIR دوران محتوى Bit Ac الى اليمين من خلال E

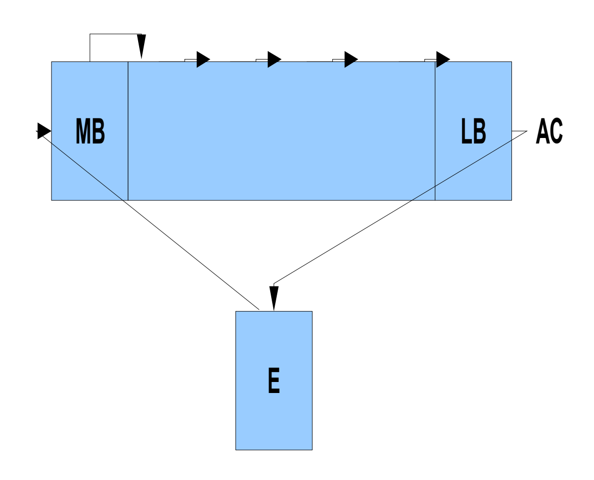
والرسم يوضح عملية الدوران عن E

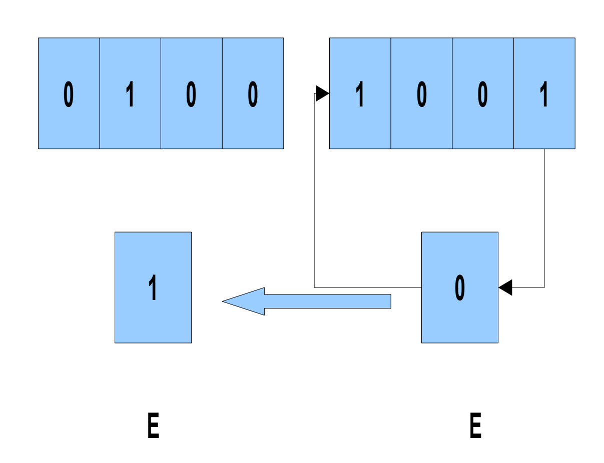
الشيفرة :

$$CIR \rightarrow 7080$$

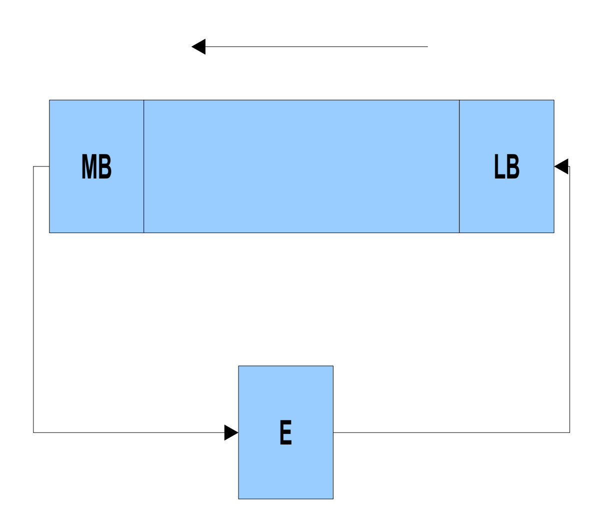
$$7080\ CIR\ RB7 : AC(15) \leftarrow E, SHR(AC), E \leftarrow AC(0)$$

نلا حظ هنا ان الشفرة تحتوى على مضمون العملية اى محتوى E يذهب الى AC وعمل تدوير لمحتوى AC الى اليسار عن طريق E وذلك لعمل SHIFT RIGHT لمحتوى ال Register ازاحة الى اليمين





6 – CIL دوران محتوى AC الى اليسار من خلال E



كما نرى ازاحة تجاه اليسار

الشفرة :

$$CIL \rightarrow 7040$$

$$7040\, CIL\ RB6 : AC(0) \leftarrow E\ , SHL(AC), E \leftarrow AC(15)$$

7 – INC اضافة واحد على AC وتخزين فى AC

الشفرة :

$$INC \rightarrow 7020$$

$$7020\, INC\ RB5 : AC \leftarrow AC + 1$$

SPA : Scab If Positive - 8

ومعنى العملية انه لو محتوى AC موجب يتم اهمال الخطوة التالية للتنفيذ اى يتم اضافة 1 على PC

اذا اولا يتم اختبار MB فى AC اذا كان 0 ----- + واذا كان 1 ----- -

الشفرة :

$$SPA \rightarrow 7010$$

$$7010\, SPA\ RB4 : IF\ AC(15) = 0\ , PC \leftarrow PC + 1$$

SNA : Scab If Negative – 9

اى اذا كانت المحتوى = 1 اى القيمة سالبة يترك الخطوة التالية للتنفيذ

الشيفرة :

$$SNA \rightarrow 7008$$

$$7008\, SNA\; RB3 : IF\; AC(15)=1\, , PC \leftarrow PC+1$$

SZA – 10 اهمال الخطوة التالية لعملية التنفيذ اذا كان محتوى AC = 0

الشفرة :

$$SZA \rightarrow 7004$$

$$7004\, SZA\; RB2 : IF\; AC=0, PC \leftarrow PC+1$$

SZE – 11 نفس العملية السابقة لو كان محتوى E = 0

الشفرة :

$$SZE \rightarrow 7002$$

$$7002\, SZE\; RB1 : IF\; E=0\, , PC \leftarrow PC+1$$

HLT – 12 خطوة توقف البرنامج عن التنفيذ وتستخدم Flip Flop تسمى S لعملية تشغيل البرنامج وايقافه

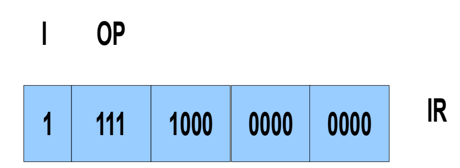
عندما تكون S = 1 البرنامج ONN وعندما تكون S = 0 البرنامج يكون OFF

الشفرة :

$$HLT \rightarrow 7001$$

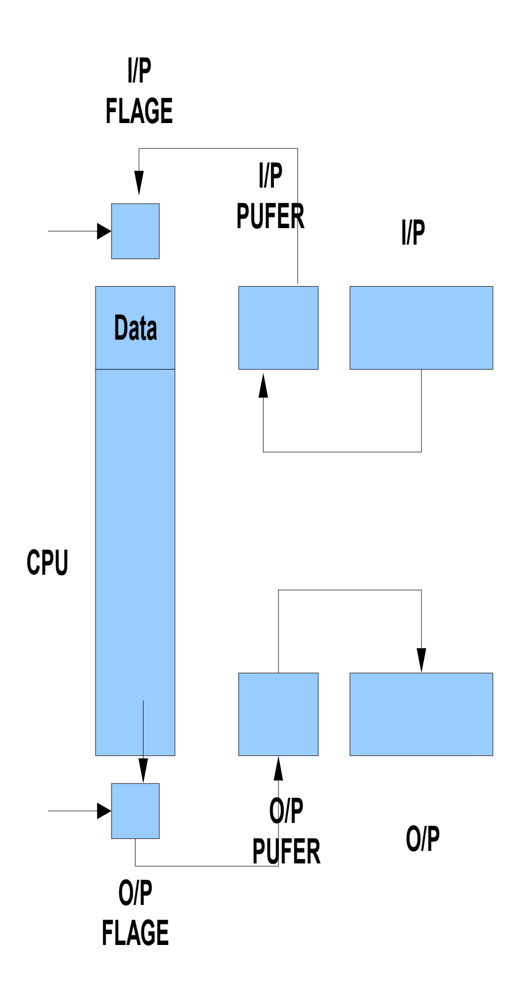
$$7001\; HLT\; RB0 : S \leftarrow 0$$

: Input – Output Instruction #



اذا بدائنا من اليسار ب 15 اذا العملية تكون Input – Output ويتولد 12 عملية مختلفة ولكن سنشرح منهم

2 فقط وذلك لاهمتهم فى الحياة العملية :



Concepts In Hard Ware

هنا نجد ان timing هو T3 وهو ثابت و ايضا D7 و I ولذلك سنعوض عنهم فى الحالتين ب P

1 – INP عملية INPUT

الشفرة :

$$F800\ INP\ PB11 : AC\ 0 \rightarrow 7\ INPUT\ REGISTER$$

هذه العملية معنها هو ادخال Bites من 0 الى 7 من AC الى Input Register

2 – OUTP عملية OUTPUT

الشفرة :

$$F400\ OUTP\ PB10 : OUTR \rightarrow AC\ 0 \rightarrow 7$$

ومعناها من 0 الى 7 داخل AC يذهب الى OUTPUT Register

نتطرق الان الى بعض العمليات الشهيرة مثل SKI وهى متى يتم تحديد قراءة Data من المفاتيح او خروجها الى الشاشة عن طريق I/P Flag , output Flag

1 – SKI اهمال خطوة تنفيذ عندما يكون I/P Flag = 1

الشفرة :

$$F200\ SKI\ PB9 : IF\,(FGI = 1) : PC \leftarrow PC + 1$$

2 – SKO اهمال خطوة لو كان O/P Flag = 1

الشفرة :

$$F100\ SKO\ PB8 : IF\,(FGO = 1),\ PC \leftarrow PC + 1$$

Concepts In Hard Ware

ملاحظات هامة :

اى برنامج ينفذ بنفس الخطوات المكتوبة ولكن فى بعض اللحظات اثناء التنفيذ توجد عملية اخرى يجب تنفيذها خارج البرنامج اذا كيف يتم ترك البرنامج وتنفيذ العملية والرجوع اليه مرة اخرى

يتم عمل Signal تسمى Incrupt وتوجد عمليتين من هذا النوع :

1 – ION : Incrupt On

ومعناها تنفيذ العملية الخارجية

الشفرة :

$$F080\ ION\ PB7 : IEN \leftarrow 1$$

ومعنى IEN اى Flag Enable

2 – IOF : Incrupt Off

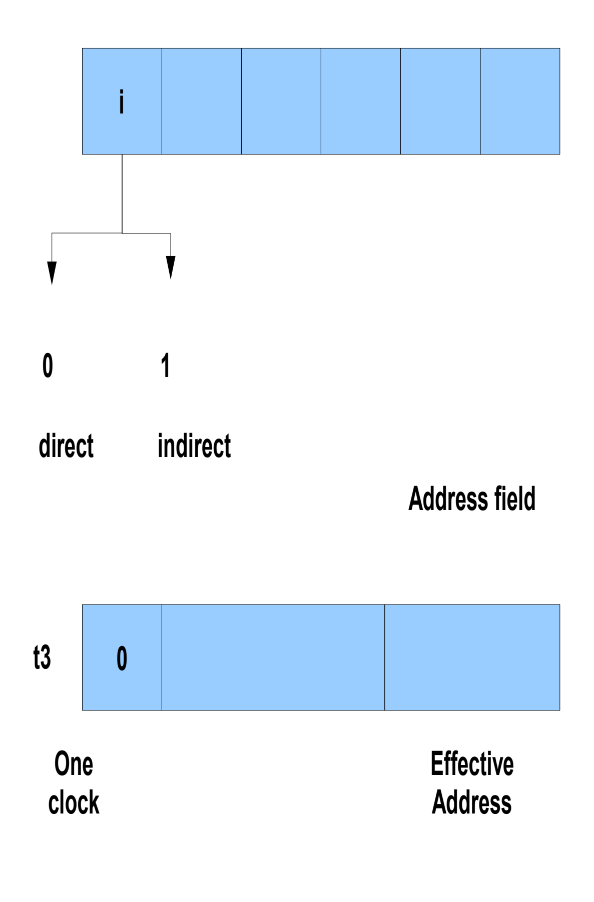
وعناها انهاء العلمية الخارجية واستكمال البرنامج الاساسى

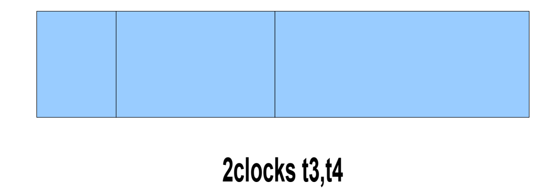
$$F040\ IOF\ PB6 : IEN \leftarrow 0$$

ومعنى IEN = 0 اى Flag Disable

Concepts In Hard Ware

- : Memory Reference Instruction #





Concepts In Hard Ware

نجد امه فى العملية الاولى Direct يتم اهمال t3 وتشغيل t4 لقراءة Address و t5 للتنفيذ

ولكن فى العملية الثانية Indirect نجد ان t3 تشير الى عنوان مكان و t4 تشير الى عنوان البيانات و t5 للتنفيذ

وبالتالى يكون t4 لقراءة البيانات و t5 للتنفيذ فى كلا الحالتين

اذا يوجد لدينا 2clocks واحدة للقراءة وواحدة للتنفيذ

الاوامر الهامة فى هذه الجزئية :

AND TO AC – 1

الشفرة :

$$T4\, D4 : DR \leftarrow M[AR]$$

$$T5\, D4 : AC \leftarrow AC \wedge DR\, , SC \leftarrow 0$$

ومعناها عند T4 يتم انتقال البيانات من الذاكرة الى DR وعند T5 يتم اضافة البيانات التى فى DR على AC وانتقالها الى AC

ADD TO AC – 2

الشفرة :

$$D4\, T4 : DR \leftarrow M[AR]$$

$$D4\, T5 : AC \leftarrow AC + DR\, , E \leftarrow COUT\, , SC \leftarrow 0$$

LDA : Load To AC – 3

الشفرة :

$$D4\, T4 : DR \leftarrow M[AR]$$

$$D4\, T5 : AC \leftarrow DR\, , SC \leftarrow 0$$

STA : Store To AC – 4

الشفرة :

$$D3\,T4 : M[AR] \leftarrow AC\,,\,SC \leftarrow 0$$

BUN : Branch Unconditionally – 5

الشفرة :

$$D4\,T4 : PC \leftarrow AR\,,\,SC \leftarrow 0$$

BSA : Branch And Save Return Address – 6

الشفرة :

$$M[AR] \leftarrow PC\,,\,PC \leftarrow AR + 1$$

ISZ : Increment And Skip If Zero – 7

الشفرة :

$$D6T4 : DR \leftarrow M[AR]$$

$$D6T5 : DR \leftarrow DR + 1\,,\,SC \leftarrow 0$$

$$D6T6 : M[AR] \leftarrow DR\,,\,IF\,(DR = 0)\,THEN\,(PC \leftarrow PC + 1\,,\,SC \leftarrow 0)$$

تم بحمد الله وتوفيقه انهاء هذا الكتاب

واعتذر عن اى خطىء او عدم وضوح بعض الكلمات وذلك نظرا لضيق الوقت

وعلى قارىء الكتاب ارسال اى استفسارات الى صاحب الكتاب

هذا الكتاب ضمن سلسة واسعة من كتب مبادىء فى الهاردوير

سيتم تن شاء الله تصميمها و ايداع تلك الكتب فى موقع الكتب العربى

السلسلة القادمة

(General Register Organization)

(Register Stack)

(Spice Programmer)

(VLSI)

(Control And Simulation)

// حاصل على شهادة – الهندسة الاليكترونية – بمنوف – جمهورية مصر العربية

البريد الاليكترونى : memorycode_84@yahoo.com

يرجى لاى استفسار على محتويات الكاتب ارسال رسالة على هذا البريد او عن طريق

المحادثة عن طريق برنامج الياهو